Sqirvy-llm

VERSION 0.0.1

I was working out some devops for a project, and I needed a simple way to make queries to LLM providers for use in a Go command line program. I didn't want to have to copypasta from a web app or a Python script. This project is an attempt to create the simplest possible Go api for making queries to LLM providers. I wanted to use Go because it is convenient to build binaries for Linux, Windows and MacOS.

GitHub Repo

API Library

This is the interface you would use to make queries to LLM providers in Go.

Most of the code was generated using Aider and the claude-3-sonnet-20240229 model. I had to do several iterations with Aider and some manual editing to get the exact code layout I wanted.

The API is in directory pkg/api. It is a very simple interface that allows you to query a provider with a prompt and get a response. It supports Anthropic, Gemini, and OpenAI providers through the 'client' interface. Here is an example of how to use the API in a command line program. Examples for the other providers are in the 'cmd' directory.

- Making a query to a provider
  - Create a new client for the provider you want to use
    - api.NewClient(api.)
    - anthropic, gemini or openai
  - Make the query with a prompt, the model name, and any options (nothing supported yet). You can request the results to be plain text or JSON
    - client.QueryText(prompt, model string, options Options) (string, error)
    - client.QueryJSON(prompt string, model string, options Options) (string, error)
  - Get the response
  - Handle any errors

```
package main

import (
    "fmt"
    "log"

    api "sqirvyllm/pkg/api"
)

func main() {
    // Create a new Anthropic client
    client, err := api.NewClient(api.Anthropic)
    if err != nil {
        log.Fatalf("Failed to create client: %v", err)
```

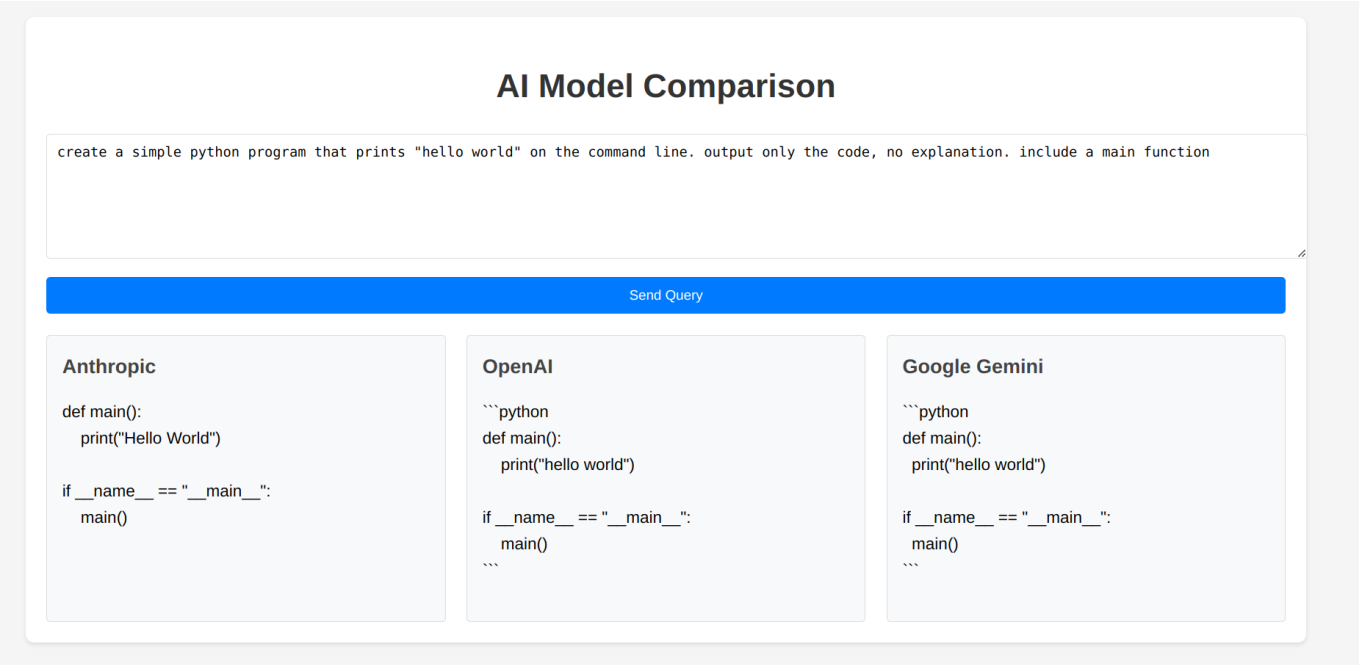
```
    }

    // Make the query with a prompt, the model name, and any options
(nothing supported yet)
    response, err := client.QueryText("say hello world", "claude-3-sonnet-
20240229", api.Options{})
    if err != nil {
        log.Fatalf("Query failed: %v", err)
    }

    fmt.Println("Response:", response)
}
```

Web App Example

Another example is in the web directory. It is a simple web app that allows you to query all three providers in parallel and compare the results. In this case ALL the code was generated using Aider and the claude-3-sonnet-20240229 model.



What Client API's Were Used

Anthropic

- github.com/anthropics/anthropic-sdk-go
- this api is a Go native client for the Anthropic API
- this api is the one recommeded by Anthropic for Go.
- It's in alpha now but seems to work without problems for these use cases.
- **The Anthropic sdk default to "ANTHROPIC_API_KEY" environment variable to authenticate**

Gemini

- "github.com/google/generative-ai-go/genai"

- "google.golang.org/api/option"
- this is the official Go client for the Gemini API supported by Google
- **The Gemini API requires a "GEMINI_API_KEY" environment variable to authenticate**

OpenAI

- OpenAI API
- Since there did not seem to be an official Go native API for OpenAI, I used the OpenAI REST API directly with the "net/http" package.
- **The OpenAI API requires a "OPENAI_API_KEY" environment variable to authenticate**